Notebook

March 9, 2025

root/final/trimodel_before_optimisation.ipynb

[1]:
```
import torch
from PIL import Image
from transformers import AutoModelForCausalLM, AutoTokenizer
import gradio as gr
import whisper  #

device = "cuda" if torch.cuda.is_available() else "cpu"

#   GLM-4V
tokenizer = AutoTokenizer.from_pretrained("/root/autodl-tmp/glm-4v-9b",␣
 ↪trust_remote_code=True)
model = AutoModelForCausalLM.from_pretrained(
    "/root/autodl-tmp/glm-4v-9b",
    torch_dtype=torch.bfloat16,
    low_cpu_mem_usage=True,
    trust_remote_code=True
).to(device).eval()

#   Whisper
whisper_model = whisper.load_model("base")  #

def transcribe_audio(audio_path):
    """          """
    if not audio_path:
        return ""

    print(f"      : {audio_path}")  # DEBUG:

    try:
        transcription = whisper_model.transcribe(audio_path)
        text_output = transcription["text"]
        return text_output if text_output.strip() else ""
    except Exception as e:
        return f"    : {str(e)}"

def generate_description(image, query):
    """      """
```

```
    if not query.strip():
        return "         "

    #    +
    if image is not None:
        image = image.convert('RGB')
        inputs = tokenizer.apply_chat_template(
            [{"role": "user", "image": image, "content": query}],
            add_generation_prompt=True,
            tokenize=True,
            return_tensors="pt",
            return_dict=True
        ).to(device)
    else:
        inputs = tokenizer.apply_chat_template(
            [{"role": "user", "content": query}],
            add_generation_prompt=True,
            tokenize=True,
            return_tensors="pt",
            return_dict=True
        ).to(device)

    gen_kwargs = {"max_length": 1000, "do_sample": True, "top_k": 1}
    with torch.no_grad():
        outputs = model.generate(**inputs, **gen_kwargs)
        outputs = outputs[:, inputs['input_ids'].shape[1]:]
        description = tokenizer.decode(outputs[0], skip_special_tokens=True)

    return description

def update_query_from_audio(audio):
    """        `   `    """
    transcribed_text = transcribe_audio(audio)
    return transcribed_text  #       UI

def gradio_interface(image, transcribed_text, query):
    """      query   `   ` + `   `    """
    final_query = query.strip() or transcribed_text.strip()  #

    if not final_query:
        return "         "

    description = generate_description(image, final_query)
    return description

# Gradio
with gr.Blocks() as interface:
```

```
    gr.Markdown("## GLM-4V   +   +       ")
    gr.Markdown("           AI      ")

    with gr.Row():
        image_input = gr.Image(label="     ", type="pil")

    with gr.Row():
        audio_input = gr.Audio(type="filepath", label="     ")
        transcribed_text = gr.Textbox(label="       ", interactive=True)  #

    with gr.Row():
        query_input = gr.Textbox(label="      ", interactive=True)  #
        submit_button = gr.Button(" ")

    output_text = gr.Textbox(label="   ")

    #         `   `
    audio_input.change(update_query_from_audio, inputs=[audio_input],␣
↪outputs=[transcribed_text])

    #      `   ` + `   `
    submit_button.click(
        gradio_interface,
        inputs=[image_input, transcribed_text, query_input],
        outputs=[output_text]
    )

interface.launch(share=True)
```

```
Loading checkpoint shards:   0%|          | 0/15 [00:00<?, ?it/s]
```

```
/root/miniconda3/lib/python3.12/site-packages/whisper/__init__.py:150:
FutureWarning: You are using `torch.load` with `weights_only=False` (the current
default value), which uses the default pickle module implicitly. It is possible
to construct malicious pickle data which will execute arbitrary code during
unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for
more details). In a future release, the default value for `weights_only` will be
flipped to `True`. This limits the functions that could be executed during
unpickling. Arbitrary objects will no longer be allowed to be loaded via this
mode unless they are explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control of the
loaded file. Please open an issue on GitHub for any issues related to this
experimental feature.
  checkpoint = torch.load(fp, map_location=device)
```

```
Running on local URL:  http://127.0.0.1:7860
```

```
Thanks for being a Gradio user! If you have questions or feedback, please join
our Discord server and chat with us: https://discord.gg/feTf9x3ZSB
Running on public URL: https://4041d8f124070fcd45.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU
upgrades, run `gradio deploy` from Terminal to deploy to Spaces
(https://huggingface.co/spaces)
```

```
<IPython.core.display.HTML object>
```

[1]:

This notebook was converted with convert.ploomber.io